PRINCIPLES OF MODELING FOR CYBER PHYSICAL SYSTEMS

Assignment #4

Parameter estimation and model evaluation

Due Date: 10-10-2019

Madhur Behl
madhur.behl@virginia.edu

ESTIMATING THE MODEL PARAMETERS

By the end of this assignment, you will have a state-space model of a single zone in EnergyPlus.
From previous assignments, you should have the following:

- `construction.m` - Computes the nominal values of the model parameters from the material characteristics in IDF.
- `model_structure.m` - The function which describes how the elements of the A,B,C,and D matrices depend non-linearly on the parameter vector 'p' in MATLAB.
- A csv file generated from EnergyPlus with the model input and output data. (This is also provided for the reference model from Assignments 2 and 3)

So far, you have successfully managed to create, and encode the state-space model structure in `MATLAB`. However, this is just half of the effort. The other half is to tune the model parameters (R,C values) to match the operation data from the zone. Remember that our goal with model based design in this case is to produce a dynamical state-system model, which can predict the zone's temperature evolution, given a forecast of disturbances and inputs.
In this worksheet, we will:

Problem 4-1: Use non-linear parameter estimation algorithm to estimate the R and C parameter values using building's data.,

Problem 4-2: Write a function which calculates the predicted temperature of the zone from input data, for a given value of the parameter vector, and

Problem 4-3: Evaluate the prediction accuracy of the model.

The following files are available on the course website on UVA Collab.

1. `5ZoneAirCooled_altmod.csv` - EnergyPlus csv data dump.
2. `data.mat` - Data file for model training and testing.
3. `parameter_estimation.m` - Main program for parameter estimation and model evaluation.
4. `getlabel.m` - Generate response predictions for a given set of parameter values.
5. `construction.m` - Solution file for Worksheet 3 based on the model structure discussed in the lectures. You are encouraged to use your own construction.m file that you submitted in Worksheet 3.
6. `model_structure.m` - Solution file for Worksheet 3 based on the model structure discussed in the lectures. You are encouraged to use your own version that you submitted in Worksheet 3.

1 PROBLEM 4-1: PARAMETER ESTIMATION

As discussed in the lectures, the parameter estimation problem for our state space model is non-linear with respect to the parameters of the model (i.e. the elements of the A,B,C, and D matrices depend non linearly on the RC parameters). Therefore, we need to rely on non-linear least squares estimation algorithms to search for the optimal values of the parameters. Least squares for parameter estimation, is the problem of finding a parameter vector θ that is a local minimizer to a function that is a sum of squared errors, possibly subject to some constraints:

$$\text{minimize}_{\theta} \sum_{i=1}^{N} (y_{true} - \hat{y}_{predict})^2$$

subject to:

$$x(i+1) = A(\theta)x(i) + B(\theta)u(i)$$
$$y(i) = C(\theta)c(i) + D(\theta)u(i)$$
$$x_0 = x_{init}$$
$$lb <= \theta <= ub$$

The notation in the optimization problem above explicitly denotes the dependence of $A(\theta)$, $B(\theta)$, $C(\theta)$, and $D(\theta)$ matrices on the parameter vector θ. In fact, you encoded this dependence in MATLAB in the previous exercise in the function `model_structure.m`

1.1 LEVENBERG-MARQUARDT AND TRUST-REGION-REFLECTIVE ALGORITHMS

Nonlinear least squares methods iteratively reduce the sum of the squares of the errors between the predicted function output and the measured data points through a sequence of updates to parameter values. The Levenberg-Marquardt curve-fitting method is a combination of two minimization methods: the gradient descent method and the Gauss-Newton method. In the gradient descent method, the sum of the squared errors is reduced by updating the parameters in the steepest-descent direction. In the Gauss-Newton method, the sum of the squared errors is reduced by assuming the least squares function is locally quadratic, and finding the minimum of the quadratic (and utilizing the Jacobian approximation to the Hessian) The Levenberg-Marquardt method acts more like a gradient-descent method when the parameters are far from their optimal value, and acts more like the Gauss-Newton method when the parameters are close to their optimal value. It belongs to a class of trust region algorithms for non-linear optimization.
In `MATLAB`, several optimization solvers are available for solving the non-linear least squares problem:

- `lsqnonlin`
- `nlinfit`
- `lsqcurvefit`

They either use the Levenberg-Marquardt or a Trust-Region-Reflective estimation algorithm.

1.2 MATLAB IMPLEMENTATION

A `parameter_estimation.m` `MATLAB` program template has been provided. This `MATLAB` program, performs the following tasks:

- Obtains the nominal parameters from `construction.m` - From Worksheet 3.

```
%% Construction Details and calclulation of the nominal values.

%Obtain nominal parameters structure and some zone construction ...
    information
[paranom,surfaces]=construction(); % NO input arguments should be ...
    provided
```

- Loads the input-output data `data.mat` from the working directory. - This mat file is provided on the course website.

```
%% Load the training dataset.
%{
(1) Read and store training data
(2) partition data into two modes: cooling ON and cooling OFF: ...
    The indices
```

```
derived above will be used here.

More about the training data:

Column# | Input data stream of interest.
3   |   Ground temperature (C)
2   |   Outside ambient temp (c)
4   |   Total internal radiative gain (W)
6   |   Total internal convective gain (W)
14  |   solar radiation on external wall (W)
10  |   solar radiation transmitted through window +door (W)
9   |   other equipment (W)
23  |   SPACE2-1 temp (C)
25  |   SPACE3-1 temp (C)
26  |   SPACE4-1 temp (C)
27  |   SPACE5-1 temp (C)
22  |   plenum temp (C)
%}

%alldata = readtrainingdata('5ZoneAirCooled_altmod.csv');
load('alldata.mat');
alldata(1,:)=[];
alldata(:,12) = (alldata(:,12)+alldata(:,20)+alldata(:,21))/3; % ...
    average indoor temperatures of all neighboring zones


% Load date/time schedule info:
% This is usedful for plotting as well as separating weekend and ...
    weekday
% schedules.
load('training_dates.mat');
dateinfo=datenum(training_dates(1:end));
```

- Creates the training and testing data-sets from the input-output data.

```
% split into training and testing datasets.
% specify the fraction of data to be used for model training ...
    (parameter
% estimation)
ntrain = floor(0.7*ndata);
ntest = ndata - ntrain;

% form the training data structure
U=zeros(ntrain,7);

U(:,1)=alldata(1:ntrain,2);     % ta: outside ambient temperature.
U(:,2)=alldata(1:ntrain,3);     % tg: ground slab temp.
U(:,3)=alldata(1:ntrain,22);    % tp: plenum temperature.
U(:,4)=alldata(1:ntrain,12);    % <tiw>: Average indoor ...
    interaction temp.
U(:,5)=alldata(1:ntrain,15);    % qsole: External solar irradiation.
U(:,6)=alldata(1:ntrain,8);     % qgain: Internal heat gains.
```

```
16 U(:,7)=alldata(1:ntrain,9);      % qcool: Cooling rate.
17
18 Y=alldata(1:ntrain,25);          % SPACE3-1 temp (output)
```

- Specifies the option-set for a multi-start non-linear least squares regression problem - *You will complete this step.*
- Returns the estimated set of parameters, and a vector of predicted values of the output (zone temperature). - *You will complete this step.*
- Evaluates the model: *You will complete this step.*

Figure 1.1 shows an example of how the estimation function `nlfit` in `MATLAB` can be used for the parameter estimation. You need not use the exact same function.

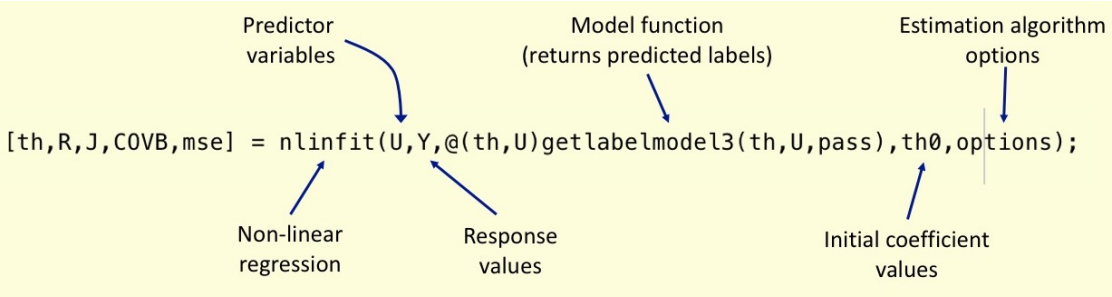

Figure 1.1: Example of a non-linear estimation function in matlab.

For non-linear parameter estimation algorithms, you need to provide a model function to generate a vector of predicted responses in order to evaluate the least squares objective function. You will learn about writing this function next.

2 PROBLEM 4-2: OBTAINING PREDICTING RESPONSE VALUES

As described earlier, the non-linear parameter estimation is essentially a search over the parameter space. For the estimation algorithm to converge to a minima we need to compute the least squares cost function:

$$\sum_{i=1}^{N} (y_{true} - \hat{y}_{predict})^2$$

which requires constructing the the prediction response vector $\hat{y}_{predict}$ (of length equal to N) for a given point in the parameter space.
You can use the `getlabel.m` template function for this part of the worksheet.
Given a parameter vector, the `getlabel` function will return a vector of predicted responses (y) of length equal to the number of samples in the training data.
Fill in the missing code to compute the vector of predicted labels from the model, with A, B, C, and D matrices computed using a given parameter vector.

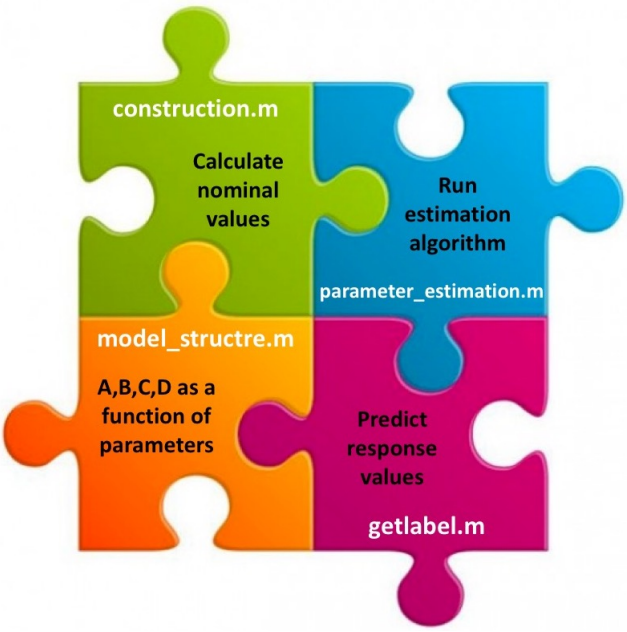

Figure 2.1: All the elements of a parameter estimation problem.

3 PROBLEM 4-3: MODEL EVALUATION

We now have all the missing pieces of the parameter estimation puzzle (Figure 2.1):

- The `construciton.m` function computes the nominal values of the parameters.
- The `model_structure.m` function specifies the elements of the state-space models A, B, C, and D in terms of the parameter vector.
- The `getlabel.m` function returns the vector of predicted responses, for a given estimate of the parameter values.
- The script `parameter_estimation.m` loads the input data, constructs the training and the test data, estimates the parameters, and evaluates the model accuracy.

Based on which estimation algorithm you chose to implement and your model structure, you can obtain an estimate of the parameter vector (with R and C values) which has the least squared error on the training data. It is recommended that you play around with different options and thresholds for the estimation algorithm and see their effect on the converged parameter values.
However, the real test for any model is how well the model generalizes on a data-set that it has not 'seen' before i.e. how well can it predict the model output on an unseen input data-set.
There are several ways to validate the output of the model. A very common way is to obtain the predictions from the model, on an input data-set, and then compare these predictions with the ground truth, of what the system actually did. The question, then becomes, how do

you compare the predicted response with the ground truth values of the response variable. While the first thing you should do is to plot both the data-sets against each other, you can use several functions which compute some measure of the *goodness of fit*. A common and effective way of evaluating how well the predictions match with reality is by calculating the root mean square error (RMSE) of the data-sets. The RMSE measures the standard deviation of the error between the predicted values and the ground truth (or simply put, its the standard deviation of the residuals of the model predictions).

$$\text{RMSE} = \sqrt{\frac{\sum_{t=1}^{N}(y_{true} - \hat{y}_{predict})^2}{N}}$$

It is beneficial to also calculate the normalized root mean squared error (NRMSE) which is a dimensionless value that helps compare the RMSE with the average of the ground truth or the range of the ground truth data. It is especially useful when the scale of the response variable is too large or too small.

$$\text{NRMSE} = \frac{RMSE}{y_{true_{max}} - y_{true_{min}}} = \frac{RMSE}{\bar{y_{true}}}$$

Submit the following:(100 points)

- **[P4.1](60)** Submit you entire code base with all the functions `construction.m`, `model_structure.m`, `getlabel.m`, and `parameter_estimation.m`. Create a <your computing ID.zip> file with all the functions above and any other functions or mat files that your code may use. Ensure that:
  - Your submitted code directory runs without any errors (rank or inversion warnings due to parameter estimations are fine, but no hard errors) and displays the NRMSE values for the training and testing data-sets as defined in the `parameter_estimation_exercise.m` template,
  - your code creates at least two plots, one for the training data model accuracy and one for the test data model accuracy. You may plot other relevant variables, parameters, or states, as needed.
- **[P4.2](40)** Try the parameter estimation with and without multi-start, and report your observations (NRMSE values and test data plots) on the performance difference between the two cases.
- **[P4.3](Extra credit-30)** Instead of the input-output data provided in the `data.mat` file; use the input-output data you created in the previous worksheet. Evaluate how the model accuracy changes between the two data-sets, given that everything else (model_structure, estimation algorithm) still remains the same.